On the Asymptotics of Solving the LWE Problem Using Coded-BKW with Sieving

Qian Guo, Thomas Johansson, *Member, IEEE,*
Erik Mårtensson and Paul Stankovski Wagner, *Member, IEEE*

***Abstract*—The Learning with Errors problem (LWE) has become a central topic in recent cryptographic research. In this paper, we present a new solving algorithm combining important ideas from previous work on improving the Blum-Kalai-Wasserman (BKW) algorithm and ideas from sieving in lattices. The new algorithm is analyzed and demonstrates an improved asymptotic performance. For the Regev parameters $q = n^2$ and noise level $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$, the asymptotic complexity is $2^{0.893n}$ in the standard setting, improving on the previously best known complexity of roughly $2^{0.930n}$. The newly proposed algorithm also provides asymptotic improvements when a quantum computer is assumed or when the number of samples is limited.**

***Index Terms*—LWE, BKW, Coded-BKW, Lattice codes, Lattice sieving.**

I. Introduction

Post-quantum crypto, the area of cryptography in the presence of quantum computers, is currently a major topic in the cryptographic community. Cryptosystems based on hard problems related to lattices are currently intensively investigated, due to their possible resistance against quantum computers. The major problem in this area, upon which cryptographic primitives can be built, is the *Learning with Errors* (LWE) problem.

LWE is an important, efficient and versatile problem. One famous application of LWE is the construction of Fully Homomorphic Encryption schemes [15]–[17], [22]. A major motivation for using LWE is its connections to lattice problems, linking the difficulty of solving LWE (on average) to the difficulty of solving instances of some (worst-case) famous lattice problems. Let us state the LWE problem.

Definition 1: Let n be a positive integer, q a prime, and let $\mathcal{X}$ be an error distribution selected as the discrete Gaussian distribution on $\mathbb{Z}_q$. Fix $\mathbf{s}$ to be a secret vector in $\mathbb{Z}_q^n$, chosen according to a uniform distribution. Denote by $L_{\mathbf{s},\mathcal{X}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error $e \in \mathbb{Z}_q$ according to $\mathcal{X}$ and returning

$$(\mathbf{a}, z) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The (search) LWE problem is to find the secret vector $\mathbf{s}$ given a fixed number of samples from $L_{\mathbf{s},\mathcal{X}}$.
The definition above gives the *search* LWE problem, as the problem description asks for the recovery of the secret vector $\mathbf{s}$. Another variant is the *decision* LWE problem. In this case the problem is to distinguish between samples drawn from $L_{\mathbf{s},\mathcal{X}}$ and a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Typically, we are then interested in distinguishers with non-negligible advantage.

For the analysis of algorithms solving the LWE problem in previous work, there are essentially two different approaches. One being the approach of calculating the specific number of operations needed to solve a certain instance for a particular algorithm, and comparing specific complexity numbers. The other approach is asymptotic analysis. Solvers for the LWE problem with suitable parameters are expected to have fully exponential complexity, say bounded by 2^{cn} as n tends to infinity. Comparisons between algorithms are made by deriving the coefficient c in the asymptotic complexity expression.

A. Related Work

We list the three main approaches for solving the LWE problem in what follows. A good survey with concrete complexity considerations is [6] and for asymptotic comparisons, see [27].

The first class is the algebraic approach, which was initialized by Arora-Ge [8]. This work was further improved by Albrecht et al., using Gröbner bases [2]. Here we point out that this type of attack is mainly, asymptotically, of interest when the noise is very small. For extremely small noise the complexity can be polynomial.

The second and most commonly used approach is to rewrite the LWE problem as a lattice problem, and therefore lattice reduction algorithms [18], [46], such as sieving and enumeration can be applied. There are several possibilities when it comes to reducing the LWE problem to some hard lattice problem. One is a direct approach, writing up a lattice from the samples and then to treat the search LWE problem as a Bounded Distance Decoding (BDD) problem [36], [37]. One can also reduce the BDD problem to a UNIQUE-SVP problem [5]. Another variant is to consider the distinguishing problem in the dual lattice [40]. Lattice-based algorithms have the advantage of not using an exponential number of samples.

The third approach is the BKW-type algorithms.

This paper was presented in part at the 23rd Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2017), Hong Kong, China.

The authors are with the Department of Electrical and Information Technology, Lund University, SE-221 00 Lund, Sweden (e-mail: {qian.guo, thomas.johansson, erik.martensson, paul.stankovski}@eit.lth.se). Qian Guo is also with the Selmer Center, Department of Informatics, University of Bergen, N-5020 Bergen, Norway (e-mail: qian.guo@uib.no).

This work was supported in part by the Swedish Research Council (Grant No. 2015-04528). Qian Guo was also supported in part by the Norwegian Research Council (Grants No. 247742/070).

1) BKW variants: The BKW algorithm was originally proposed by Blum, Kalai and Wasserman [13] for solving the Learning Parity with Noise (LPN) problem (LWE for $q = 2$). It resembles Wagner's generalized birthday approach [47].

For the LPN case, there has been a number of improvements to the basic BKW approach. In [35], transform techniques were introduced to speed up the search part. Further improvements came in work by Kirchner [30], Bernstein and Lange [11], Guo et al. [23], Zhang et al. [49], Bogos and Vaudenay [14].

Albrecht et al. were first to apply BKW to the LWE problem [3], which they followed up with Lazy Modulus Switching (LMS) [4], which was further improved by Duc et al. in [21]. The basic BKW approach for LWE was improved in [25] and [31], resulting in an asymptotic improvement. These works improved by reducing a variable number of positions in each step of the BKW procedure as well as introducing a coding approach. Although the two algorithms were slightly different, they perform asymptotically the same and we refer to the approach as coded-BKW. It was proved in [31] that the asymptotic complexity for Regev parameters (public-key cryptography parameter) $q = n^2$ and noise level $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$ is $2^{0.930n+o(n)}$, the currently best known asymptotic performance for such parameters.

2) Sieving algorithms: A key part of the algorithm to be proposed is the use of sieving in lattices. The first sieving algorithm for solving the shortest vector problem was proposed by Ajtai, Kumar and Sivakumar in [1], showing that SVP can be solved in time and memory $2^{\Theta(n)}$. Subsequently, we have seen the NV-sieve [43], List-sieve [41], and provable improvement of the sieving complexity using the birthday paradox [26], [44].

With heuristic analysis, [43] started to derive a complexity of $2^{0.415n+o(n)}$, followed by GaussSieve [41], 2-level sieve [48], 3-level sieve [50] and overlattice-sieve [10]. Laarhoven started to improve the lattice sieving algorithms employing algorithmic breakthroughs in solving the nearest neighbor problem, angular LSH [32], and spherical LSH [34]. The asymptotically most efficient approach when it comes to time complexity is Locality Sensitive Filtering (LSF) [9] with both a space and time complexity of $2^{0.292n+o(n)}$. Using quantum computers, the complexity can be reduced to $2^{0.265n+o(n)}$ (see [33]) by applying Grover's quantum search algorithm.

B. Contributions

We propose a new algorithm for solving the LWE problem combining previous combinatorial methods with an important algorithmic idea – using a sieving approach. Whereas BKW combines vectors to reduce positions to zero, the previously best improvements of BKW, like coded-BKW, reduce more positions but at the price of leaving a small but in general nonzero value in reduced positions. These values are considered as additional noise. As these values increase in magnitude for each step, because we add them together, they have to be very small in the initial steps. This is the reason why in coded-BKW the number of positions reduced in a step is increasing with the step index. We have to start with a small number of reduced positions, in order to not obtain a noise that is too large.

The proposed algorithm tries to solve the problem of the growing noise from the coding part (or LMS) by using a sieving step to make sure that the noise from treated positions does not grow, but stays approximately of the same size. The basic form of the new algorithm then contains two parts in each iterative step. The first part reduces the magnitude of some particular positions by finding pairs of vectors that can be combined. The second part performs a sieving step covering all positions from all previous steps, making sure that the magnitude of the resulting vector components is roughly as in the already size-reduced part of the incoming vectors.

We analyze the new algorithm from an asymptotic perspective, proving a new improved asymptotic performance. For the asymptotic Regev parameters $q = n^2$ and noise level $\sigma = n^{1.5}$, the result is a time and space complexity of $2^{0.8951n+o(n)}$, which is a significant asymptotic improvement. We also get a first quantum acceleration (with complexity of $2^{0.8856n+o(n)}$) for the Regev parameters by using the performance of sieving in the quantum setting.

In addition, when the sample complexity is limited, e.g, to a polynomial number in n like $\Theta(n \log n)$, the new algorithm outperforms the previous best solving algorithms for a wide range of parameter choices.

Lastly, the new algorithm can be flexibly extended and generalized in various ways. We present three natural extensions further improving its asymptotic performance. For instance, by changing the reduction scale in each sieving step, we reduce the time and space complexity for the Regev parameters to $2^{0.8927n+o(n)}$ in the standard setting. With the help of quantum computers, the complexity can be even further reduced, down to $2^{0.8795n+o(n)}$.

C. Organization

The remaining parts of the paper are organized as followed. We start with some preliminaries in Section II, including more basics on LWE, discrete Guassians and sieving in lattices. In Section III we review the details of the BKW algorithm and some recent improvements. Section IV just contains a simple reformulation. In Section V we give the new algorithm in its basic form and in Section VI we derive the optimal parameter selection and perform the asymptotic analysis. In Section VII we derive asymptotic expressions for LWE with sparse secrets, and show an improved asymptotic performance when only having access to a polynomial number of samples. Section VIII contains new versions of coded-BKW with sieving, that further decrease the asymptotic complexity. Finally, we conclude the paper in Section IX.

II. Background

A. Notations

Throughout the paper, the following notations are used.

- We write $\log(\cdot)$ for the base 2 logarithm and $\ln(\cdot)$ for the natural logarithm.

- In an n-dimensional Euclidean space $\mathbb{R}^n$, by the norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ we refer to its L_2-norm, defined as
$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}.$$
We then define the Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ as $\|\mathbf{x} - \mathbf{y}\|$.
- An element in $\mathbb{Z}_q$ is represented as the corresponding value in $[-\frac{q-1}{2}, \frac{q-1}{2}]$.
- For an $[N, k_0]$ linear code, N denotes the code length and k_0 denotes the dimension.
- We use the following standard notations for asymptotic analysis.
  - $f(n) = \mathcal{O}(g(n))$ if there exists a positive constant C, s.t., $|f(n)| \leq C \cdot g(n)$ for n sufficiently large.
  - $f(n) = \Omega(g(n))$ if there exists a positive constant C, s.t., $|f(n)| \geq C \cdot g(n)$ for n sufficiently large.
  - $f(n) = \Theta(g(n))$ if there exist positive constants C_1 and C_2, s.t., $C_1 \cdot g(n) \leq |f(n)| \leq C_2 \cdot g(n)$ for n sufficiently large.
  - $f(n) = o(g(n))$ if for every positive C, we have $|f(n)| < C \cdot g(n)$ for n sufficiently large.

B. LWE Problem Description

Rather than giving a more formal definition of the decision version of LWE, we instead reformulate the search LWE problem, because our main purpose is to investigate its solving complexity. Assume that m samples
$$(\mathbf{a}_1, z_1), (\mathbf{a}_2, z_2), \ldots, (\mathbf{a}_m, z_m),$$
are drawn from the LWE distribution $L_{\mathbf{s},\mathcal{X}}$, where $\mathbf{a}_i \in \mathbb{Z}_q^n, z_i \in \mathbb{Z}_q$. Let $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m) = \mathbf{sA}$. We can then write
$$\mathbf{z} = \mathbf{sA} + \mathbf{e},$$
where $\mathbf{A} = \begin{bmatrix}\mathbf{a}_1^\intercal & \mathbf{a}_2^\intercal & \cdots & \mathbf{a}_n^\intercal\end{bmatrix}$, $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$ and $e_i \xleftarrow{\$} \mathcal{X}$. Therefore, we have reformulated the search LWE problem as a decoding problem, in which the matrix $\mathbf{A}$ serves as the generator matrix for a linear code over $\mathbb{Z}_q$ and $\mathbf{z}$ is the received word. We see that the problem of searching for the secret vector $\mathbf{s}$ is equivalent to that of finding the codeword $\mathbf{y} = \mathbf{sA}$ such that the Euclidean distance $||\mathbf{y} - \mathbf{z}||$ is minimal.

1) The Secret-Noise Transformation: An important transformation [7], [30] can be applied to ensure that the secret vector follows the same distribution $\mathcal{X}$ as the noise. The procedure works as follows. We first write $\mathbf{A}$ in systematic form via Gaussian elimination. Assume that the first n columns are linearly independent and form the matrix $\mathbf{A_0}$. We then define $\mathbf{D} = {\mathbf{A_0}}^{-1}$ and write $\hat{\mathbf{s}} = \mathbf{sD}^{-1} - (z_1, z_2, \ldots, z_n)$. Hence, we can derive an equivalent problem described by $\hat{\mathbf{A}} = (\mathbf{I}, \hat{\mathbf{a}}_{n+1}^\intercal, \hat{\mathbf{a}}_{n+2}^\intercal, \cdots, \hat{\mathbf{a}}_m^\intercal)$, where $\hat{\mathbf{A}} = \mathbf{DA}$. We compute
$$\hat{\mathbf{z}} = \mathbf{z} - (z_1, z_2, \ldots, z_n)\hat{\mathbf{A}} = (\mathbf{0}, \hat{z}_{n+1}, \hat{z}_{n+2}, \ldots, \hat{z}_m).$$
Using this transformation, one can assume that each entry in the secret vector is now distributed according to $\mathcal{X}$.

The noise distribution $\mathcal{X}$ is usually chosen as the discrete Gaussian distribution, which will be briefly discussed in Section II-C.

C. Discrete Gaussian Distribution

We start by defining the discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and variance σ^2, denoted $D_{\mathbb{Z},\sigma}$. That is, the probability distribution obtained by assigning a probability proportional to $\exp(-x^2/2\sigma^2)$ to each $x \in \mathbb{Z}$. The discrete Gaussian over $\mathbb{Z}^n$ with variance σ^2, denoted $D_{\mathbb{Z}^n,\sigma}$, is defined as the product distribution of n independent copies of $D_{\mathbb{Z},\sigma}$.

Then, the discrete Gaussian distribution $\mathcal{X}$ over $\mathbb{Z}_q$ with variance σ^2 (also denoted $\mathcal{X}_\sigma$) can be defined by folding $D_{\mathbb{Z},\sigma}$ and accumulating the value of the probability mass function over all integers in each residue class modulo q.

Following the path of previous work [3], we assume that in our discussed instances, the discrete Gaussian distribution can be approximated by the continuous counterpart. For instance, if X is drawn from $\mathcal{X}_{\sigma_1}$ and Y is drawn from $\mathcal{X}_{\sigma_2}$, then $X + Y$ is regarded as being drawn from $\mathcal{X}_{\sqrt{\sigma_1^2+\sigma_2^2}}$. This approximation, which is widely-adopted in literature (e.g., [4], [25], [28]), is motivated by the fact that a sufficient wild Gaussian[1] blurs the discrete structures.[2]

1) The sample complexity for distinguishing.: To estimate the solving complexity, we need to determine the number of required samples to distinguish between the uniform distribution on $\mathbb{Z}_q$ and $\mathcal{X}_\sigma$. Relying on standard theory from statistics, when the noise variance σ is sufficiently large to approximate the discrete Gaussian by a continuous one mod q, using either previous work [36] or Bleichenbacher's definition of bias [42], we can conclude that the required number of samples is
$$C \cdot e^{2\pi\left(\frac{\sigma\sqrt{2\pi}}{q}\right)^2}, \tag{1}$$
where C is a small positive constant.

D. Sieving in Lattices

We here give a brief introduction to the sieving idea and its application in lattices for solving the shortest vector problem (SVP). For an introduction to lattices, the SVP problem, and sieving algorithms, see e.g. [9].

In sieving, we start with a list $\mathcal{L}$ of relatively short lattice vectors. If the list size is large enough, we will obtain many pairs of $\mathbf{v}, \mathbf{w} \in \mathcal{L}$, such that $\|\mathbf{v} \pm \mathbf{w}\| \leq \max\{\|\mathbf{v}\|, \|\mathbf{w}\|\}$. After reducing the size of these lattice vectors a polynomial number of times, one can expect to find the shortest vector.

The core of sieving is thus to find a close enough neighbor $\mathbf{v} \in \mathcal{L}$ efficiently, for a vector $\mathbf{w} \in \mathcal{L}$, thereby reducing the size by further operations like addition or subtraction.

[1] It is proven in [39] that the noise is sufficiently large for the integer lattice of $\mathbb{Z}^n$ if σ is of order $\Omega(n^c)$ for $c > 0.5$.

[2] The LWE noise distribution is formally treated in [21] and in [31]. However, in [21], only the original BKW steps are investigated, and in [31], a non-standard LWE noise distribution is assumed. Both are non-applicable.

On the other hand, the adopted assumption from the literature to approximate the discrete Gaussian is strong. We can simplify it to two weaker heuristic assumptions in our deviation. Firstly, many noise variables with small variance will be generated and be added or subtracted. We assume that all the small noise variables are independent so that we can add their variance as can be done in the continuous Gaussian case. Secondly, by intuition from the central limit theorem, we assume that the final noise variable is close to a continuous Gaussian mod q. Thus, we can use the formula from [36] to estimate the required number of samples.

This is also true for our newly proposed algorithm in a later section, since by sieving we solely desire to control the size of the added/subtracted vectors. For this specific purpose, many famous probabilistic algorithms have been proposed, e.g., Locality Sensitive Hashing (LSH) [29], Bucketing coding [20], and May-Ozerov's algorithm [38] in the Hamming metric with important applications to decoding binary linear codes.

In the Euclidean metric, the state-of-the-art algorithm in the asymptotic sense is Locality Sensitive Filtering (LSF) [9], which requires $2^{0.2075n+o(n)}$ samples. In the classic setting, the time and memory requirements are both in the order of $2^{0.292n+o(n)}$. The constant hidden in the running time exponent can be reduced to 0.265 in the scenario of quantum computing. In the remaining part of the paper, we choose the LSF algorithm for the best asymptotic performance when we need to instantiate the sieving method.

III. The BKW Algorithm

The BKW algorithm is the first sub-exponential algorithm for solving the LPN problem, originally proposed by Blum, Kalai and Wasserman [12], [13]. It can also be trivially adopted to the LWE problem, with single-exponential complexity.

A. Plain BKW

The algorithm consists of two phases: the reduction phase and the solving phase. The essential improvement comes from the first phase, whose underlying fundamental idea is the same as Wagner's generalized birthday algorithm [47]. That is, using an iterative collision procedure on the columns in the matrix $\mathbf{A}$, one can reduce its row dimension step by step, and finally reach a new LWE instance with a much smaller dimension. The solving phase can then be applied to recover the secret vector. We describe the core procedure of the reduction phase, called a plain BKW step, as follows. Let us start with $\mathbf{A}_0 = \mathbf{A}$.

Dimension reduction: In the i-th iteration, we look for combinations of two columns in $\mathbf{A}_{i-1}$ that add (or subtract) to zero in the last b entries. Suppose that one finds two columns $\mathbf{a}^{\mathrm{T}}_{j_1,i-1}, \mathbf{a}^{\mathrm{T}}_{j_2,i-1}$ such that

$$\mathbf{a}_{j_1,i-1} \pm \mathbf{a}_{j_2,i-1} = [* \quad * \quad \cdots \quad * \quad \underbrace{0 \quad 0 \quad \cdots \quad 0}_{b \text{ symbols}}],$$

where $*$ means any value. We then generate a new vector $\mathbf{a}_{j,i} = \mathbf{a}_{j_1,i-1} \pm \mathbf{a}_{j_2,i-1}$. We obtain a new generator matrix $\mathbf{A}_i$ for the next iteration, with its dimension reduced by b, if we remove the last b all-zero positions with no impact on the output of the inner product operation. We also derive a new "observed symbol" as $z_{j,i} = z_{j_1,i-1} \pm z_{j_2,i-1}$.

A trade-off: After one step of this procedure, we can see that the new noise variable is $e_{j,i} = e_{j_1,i-1} \pm e_{j_2,i-1}$. If the noise variables $e_{j_1,i-1}$ and $e_{j_2,i-1}$ both follow the Gaussian distribution with variance σ^2_{i-1}, then the new noise variable $e_{j,i}$ is considered Gaussian distributed with variance $\sigma^2_i = 2\sigma^2_{i-1}$.

After t_0 iterations, we have reduced the dimension of the problem to $n - t_0 b$. The final noise variable is thus a summation of 2^{t_0} noise variables generated from the LWE oracle. We therefore know that the noise connected to each column is of the form

$$e = \sum_{j=1}^{2^{t_0}} e_{i_j},$$

and the total noise is approximately Gaussian with variance $2^{t_0} \cdot \sigma^2$.

The remaining solving phase is to solve this transformed LWE instance. This phase does not affect its asymptotic complexity but has significant impact on its actual running time for concrete instances.

Similar to the original proposal [13] for solving LPN, which recovers 1 bit in the secret vector via majority voting, Albrecht et al. [3] exhaust one secret entry using a distinguisher. The complexity is further reduced by Duc et al. [21] using Fast Fourier Transform (FFT) to recover several secret entries simultaneously.

B. Coded-BKW

As described above, in each BKW step, we try to collide a large number of vectors $\mathbf{a}_i$ in a set of positions denoted by an index set I. We denote this sub-vector of a vector $\mathbf{a}$ as $\mathbf{a}_I$. We set the size[3] of the collision set to $\frac{q^b-1}{2}$, a very important parameter indicating the final complexity of the algorithm.

In this part we describe another idea that, instead of zeroing out the vector $\mathbf{a}_I$ by collisions, we try to collide vectors to make $\mathbf{a}_I$ small. The advantage of this idea is that one can handle more positions in one step for the same size of the collision set.

This idea was first formulated by Albrecht et al. in PKC 2014 [4], aiming for solving the LWE problem with a small secret. They proposed a new technique called Lazy Modulus Switching (LMS). Then, in CRYPTO 2015, two new algorithms with similar underlying algorithmic ideas were proposed independently in [25] and [31], highly enhancing the performance in the sense of both asymptotic and concrete complexity. Using the secret-noise transformation, these new algorithms can be used to solve the standard LWE problem.

In this part we use the notation from [25] to describe the BKW variant called coded-BKW, as it has the best concrete performance, i.e., it can reduce the magnitude of the noise by a constant factor compared with its counterpart technique LMS. The core step – the coded-BKW step – can be described as follows.

Considering step i in the reduction phase, we choose a q-ary $[n_i, b]$ linear code, denoted $\mathcal{C}_i$, that can be employed to construct a lattice code, e.g., using Construction A (see [19] for details). The sub-vector $\mathbf{a}_I$ can then be written in terms of its two constituents, the codeword part $\mathbf{c}_I \in \mathcal{C}_i$ and an error part $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$. That is,

$$\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I. \tag{2}$$

[3]Naively the size is q^b. However, using the fact that vectors with subsets $\mathbf{a}_I$ and $-\mathbf{a}_I$ can be mapped into the same category, we can reduce the size to $\frac{q^b-1}{2}$. The zero vector gets its own category.

We rewrite the inner product $\langle \mathbf{s}_I, \mathbf{a}_I \rangle$ as

$$\langle \mathbf{s}_I, \mathbf{a}_I \rangle = \langle \mathbf{s}_I, \mathbf{c}_I \rangle + \langle \mathbf{s}_I, \mathbf{e}_I \rangle .$$

We can cancel out the part $\langle \mathbf{s}_I, \mathbf{c}_I \rangle$ by subtracting two vectors mapped to the same codeword, and the remaining difference is the noise. Using the same kind of reasoning, the size of the collision set can be $\frac{q^b-1}{2}$, as in the plain BKW step.

If we remove n_i positions in the i-th step, then we have removed $\sum_{i=1}^{t} n_i$ positions ($n_i \geq b$) in total. Thus, after guessing the remaining secret symbols in the solving phase, we need to distinguish between the uniform distribution and the distribution representing a sum of noise variables, i.e.,

$$z = \sum_{j=1}^{2^t} e_{i_j} + \sum_{i=1}^{n} s_i (E_i^{(1)} + E_i^{(2)} + \cdots + E_i^{(t)}), \qquad (3)$$

where $E_i^{(h)} = \sum_{j=1}^{2^{t-h+1}} \hat{e}_{i_j}^{(h)}$ and $\hat{e}_{i_j}^{(h)}$ is the noise introduced in the h-th coded-BKW step. Here at most one error term $E_i^{(h)}$ is non-zero for one position in the index set, and the overall noise can be estimated according to Equation (3).

The remaining problem is to analyze the noise level introduced by coding. In [25], it is assumed that every $E_i^{(h)}$ is close to a Gaussian distribution, which is tested in implementation. Based on known results (e.g., [19]) on lattice codes, in [25], the standard deviation σ introduced by employing a q-ary $[N, k]$ linear code is estimated by

$$\sigma \approx q^{1-k/N} \cdot \sqrt{G(\Lambda_{N,k})}, \qquad (4)$$

where $G(\Lambda_{N,k})$ is a code-related parameter satisfying

$$\frac{1}{2\pi e} < G(\Lambda_{N,k}) \leq \frac{1}{12}.$$

In [25], the chosen codes are with varying rates to ensure that the noise contribution of each position is equal. This is principally similar to the operation of changing the modulus size in each reduction step in [31]. It is trivial to get a code with $G(L_{N,k})$ larger than $1/12$. While choosing a better code is important for concrete complexity, for asymptotic complexity this trivial code is as fast as any other code.

IV. A Reformulation

Let us reformulate the LWE problem and the steps in the different algorithms in a matrix form. Recall that we have the LWE samples in the form $\mathbf{z} = \mathbf{sA} + \mathbf{e}$. We write this as

$$(\mathbf{s}, \mathbf{e}) \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix} = \mathbf{z}. \qquad (5)$$

The entries in the unknown left-hand side vector $(\mathbf{s}, \mathbf{e})$ are all i.i.d. The matrix above is denoted as $\mathbf{H_0} = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix}$ and it is a known quantity, as well as $\mathbf{z}$.

By multiplying Equation (5) from the right with special matrices $\mathbf{P_i}$ we are going to reduce the size of columns in the matrix. Starting with

$$(\mathbf{s}, \mathbf{e})\mathbf{H_0} = \mathbf{z},$$

we find a matrix $\mathbf{P_0}$ and form $\mathbf{H_1} = \mathbf{H_0P_0}$, $\mathbf{z}_1 = \mathbf{zP_0}$, resulting in

$$(\mathbf{s}, \mathbf{e})\mathbf{H_1} = \mathbf{z}_1.$$

Continuing this process for t steps, we have formed $\mathbf{H_t} = \mathbf{H_0P_0} \cdots \mathbf{P_{t-1}}$, $\mathbf{z}_t = \mathbf{zP_0} \cdots \mathbf{P_{t-1}}$.

Here the dimensionality of the $\mathbf{P_i}$ matrices depends on how the total number of samples changes in each step. If we keep the total number of samples constant, then all $\mathbf{P_i}$ matrices have size $m \times m$.

Plain BKW can be described as each $\mathbf{P_i}$ having columns with only two nonzero entries, both from the set $\{-1, 1\}$. The BKW procedure subsequently cancels rows in the $\mathbf{H_i}$ matrices in a way such that $\mathbf{H_t} = \begin{pmatrix} \mathbf{0} \\ \mathbf{H'_t} \end{pmatrix}$, where columns of $\mathbf{H'_t}$ have 2^t non-zero entries[4]. The goal is to minimize the magnitude of the column entries in $\mathbf{H_t}$. The smaller magnitude, the larger advantage in the corresponding samples.

The improved techniques like LMS and coded-BKW reduce the $\mathbf{H_t}$ similar to the BKW, but improves by using the fact that the top rows of $\mathbf{H_t}$ do not have to be canceled to $\mathbf{0}$. Instead, entries are allowed to be of the same norm as in the $\mathbf{H'_t}$ matrix.

V. A BKW-Sieving Algorithm for the LWE Problem

The algorithm we propose uses a similar structure as the coded-BKW algorithm. The new idea involves changing the BKW step to also include a sieving step. In this section we give the algorithm in a simple form, allowing for some asymptotic analysis. We exclude some steps that give non-asymptotic improvements. We assume that each entry in the secret vector $\mathbf{s}$ is distributed according to $\mathcal{X}$.

Algorithm 1 Coded-BKW with Sieving (main steps)

Input: Matrix $\mathbf{A}$ with n rows and m columns, received vector $\mathbf{z}$ of length m and algorithm parameters $t, n_i, 1 \leq i \leq t$, B

```
change the distribution of the secret vector (Gaussian elimination)
for i from 1 to t do:
    for all columns h ∈ H_{i-1} do:
        Δ = CodeMap(h, i)
        put h in list L_Δ
    for all lists L_Δ do:
        S_Δ = Sieve(L_Δ, i, √N_i · B)
        put all S_Δ as columns in H_i
guess the s_n entry using hypothesis testing
```

A summary of coded-BKW with sieving is detailed in Algorithm 1.

Note that one may also use some advanced distinguisher, e.g., the FFT distinguisher, which is important to the concrete complexity, but not for the asymptotic performance.

[4]Sometimes we get a little fewer than 2^t entries since 1s can overlap. However, this probability is low and does not change the analysis.

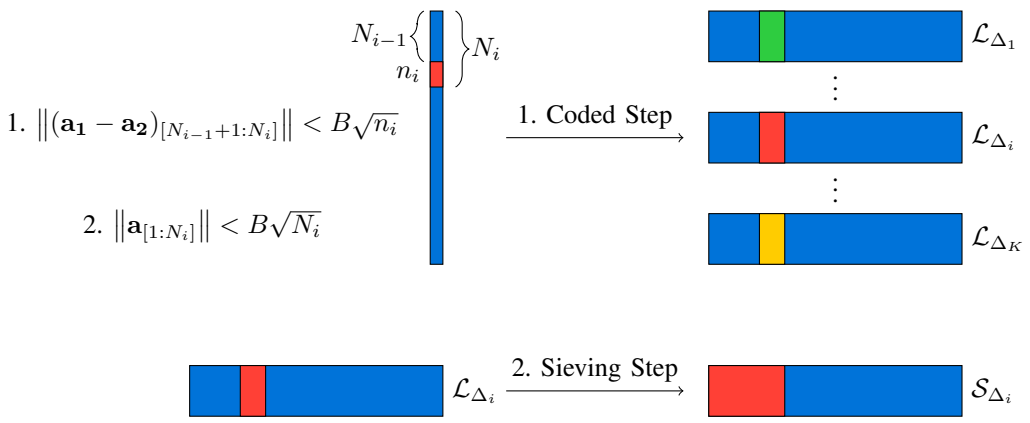

Fig. 1. A micro picture of how one step of coded-BKW with sieving works.

A. Initial Guessing Step

We select a few entries of $\mathbf{s}$ and guess these values (according to $\mathcal{X}$). We run through all likely values and for each of them we do the steps below. Based on a particular guess, the sample equations need to be rewritten accordingly.

For simplicity, the remaining unknown values are still denoted $\mathbf{s}$ after this guessing step and the length of $\mathbf{s}$ is still denoted n.

B. Transformation Steps

We start with some simplifying notation. The n positions in columns in $\mathbf{A}$ (first n positions in columns of $\mathbf{H}$) are considered as a concatenation of smaller vectors. We assume that these vectors have lengths which are $n_1, n_2, n_3, \ldots, n_t$, respectively, in such a way that $\sum_{i=1}^{t} n_i = n$. Also, let $N_j = \sum_{i=1}^{j} n_i$, for $j = 1, 2, \ldots, t$.

Before explaining the algorithmic steps, we introduce two notations that will be used later.

Notation CodeMap$(\mathbf{h}, i)$: We assume, following the idea of coded-BKW, that we have fixed a lattice code $\mathcal{C}_i$ of length n_i. The vector $\mathbf{h}$ fed as input to CodeMap is first considered only restricted to the positions $N_{i-1}+1$ to N_i, i.e., as a vector of length n_i. This vector, denoted $\mathbf{h}_{[N_{i-1}+1,N_i]}$, is then mapped to the closest codeword in $\mathcal{C}_i$. This closest codeword is denoted CodeMap$(\mathbf{h}, i)$.

The code $\mathcal{C}_i$ needs to have an associated procedure of quickly finding the closest codeword for any given vector. One could then use a simple code or a more advanced code. From an asymptotic viewpoint, it does not matter, but in a practical implementation there can be a difference. We are going to select the parameters in such a way that the distance to the closest codeword is expected to be no more than $\sqrt{n_i} \cdot B$, where B is a constant.

Notation Sieve$(\mathcal{L}_\Delta, i, \sqrt{N_i} \cdot B)$: The input $\mathcal{L}_\Delta$ contains a list of vectors. We are only considering them restricted to the first N_i positions. This procedure will find differences between any two vectors such that the norm of the difference restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. All such differences are put in a list $\mathcal{S}_\Delta$ which is the output of the procedure. On average, the list $\mathcal{S}_\Delta$ should have roughly the same amount of vectors as $\mathcal{L}_\Delta$.

We assume that the vectors in the list $\mathcal{L}_\Delta$ restricted to the first N_i positions, all have a norm of about $\sqrt{N_i} \cdot B$. Then the problem is solved by algorithms for sieving in lattices, for example using Locality-Sensitive Hashing/Filtering.

For the description of the main algorithm, recall that

$$(\mathbf{s}, \mathbf{e})\mathbf{H_0} = \mathbf{z},$$

where $\mathbf{H_0} = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix}$. We are going to perform t steps to transform $\mathbf{H_0}$ into $\mathbf{H_t}$ such that the columns in $\mathbf{H_t}$ are "small". Again, we look at the first n positions in a column corresponding to the $\mathbf{A}$ matrix. Since we are only adding or subtracting columns using coefficients in $\{-1, 1\}$, the remaining positions in the column are assumed to contain 2^i nonzero positions either containing a -1 or a 1, after i steps[5].

C. A BKW-Sieving Step

We are now going to fix an average level of "smallness" for a position, which is a constant denoted B, as above. The idea of the algorithm is to keep the norm of considered vectors of some length n' below $\sqrt{n'} \cdot B$.

A column $\mathbf{h} \in \mathbf{H_0}$ will now be processed by first computing $\Delta = \text{CodeMap}(\mathbf{h}, 1)$. Then we place $\mathbf{h}$ in the list $\mathcal{L}_\Delta$. After running through all columns $\mathbf{h} \in \mathbf{H_0}$ they have been sorted into lists $\mathcal{L}_\Delta$. Use K to denote the total number of lists.

We then run through the lists, each containing roughly m/K columns. We perform a sieving step, according to $\mathcal{S}_\Delta = \text{Sieve}(\mathcal{L}_\Delta, \sqrt{N_1} \cdot B)$, for all $\Delta \in \mathcal{C}_i$. The result is a list of vectors, where the norm of each vector restricted to the first N_1 positions is less than $\sqrt{N_1} \cdot B$. The indices of any , i_j, i_k are kept in such a way that we can compute a new received symbol $z = z_{i_j} - z_{i_k}$. All vectors in all lists $\mathcal{S}_\Delta$ are now put as columns in $\mathbf{H_1}$. We now have a matrix $\mathbf{H_1}$ where the norm of each column restricted to the first n_1 positions is less than $\sqrt{N_1} \cdot B$. This is the end of the first step.

Next, we repeat roughly the same procedure another $t-1$ times. A column $\mathbf{h} \in \mathbf{H_{i-1}}$ will now be processed by first computing $\Delta = \text{CodeMap}(\mathbf{h}, i)$. We place $\mathbf{h}$ in the list $\mathcal{L}_\Delta$.

[5]Sometimes we get a little fewer than 2^t entries since 1s can overlap. However, this probability is low and does not change the analysis.

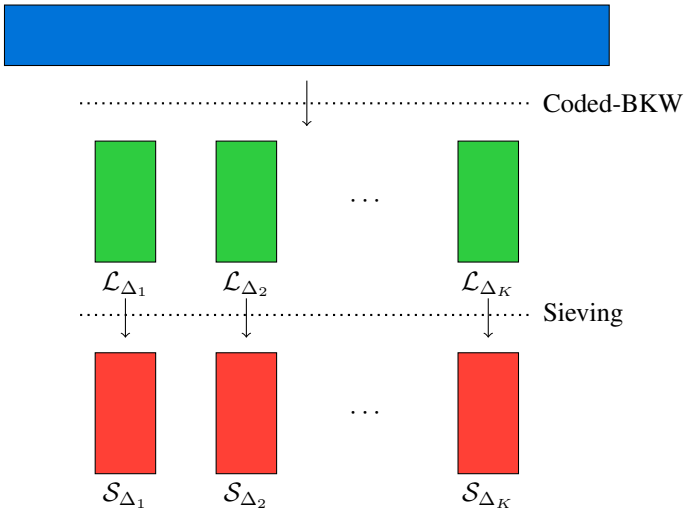

Fig. 2. A macro picture of how one step of coded-BKW with sieving works. The set of lists $\mathcal{S}_{\Delta_1}, \ldots, \mathcal{S}_{\Delta_K}$ constitutes the samples for the next step of coded-BKW with sieving.

After running through all columns $\mathbf{h} \in \mathbf{H_{i-1}}$ they have been sorted in K lists $\mathcal{L}_\Delta$.

We run through all lists, where each list contains roughly m/K columns. We perform a sieving step, according to $\mathcal{S}_\Delta = \text{Sieve}(\mathcal{L}_\Delta, i, \sqrt{N_i} \cdot B)$. The result is a list of vectors where the norm of each vector restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. A new received symbol is computed. All vectors in all lists $\mathcal{S}_\Delta$ are now put as columns in $\mathbf{H_i}$. We get a matrix $\mathbf{H_i}$ where the norm of each column restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. This is repeated for $i = 2, \ldots, t$. We assume that the parameters have been chosen in such a way that each matrix $\mathbf{H_i}$ can have m columns.

In each step i, we make the heuristic assumption that the vectors of $\mathcal{L}_\Delta$ are uniformly distributed on a sphere with radius $\sqrt{N_i} \cdot B$. This is a standard assumption[6] in papers on lattice sieving.

After performing these t steps we end up with a matrix $\mathbf{H_t}$ such that the norm of columns restricted to the first n positions is bounded by $\sqrt{n} \cdot B$ and the norm of the last m positions is roughly $2^{t/2}$. Altogether, this should result in samples generated as

$$\mathbf{z} = (\mathbf{s}, \mathbf{e})\mathbf{H_t}.$$

The values in the $\mathbf{z}$ vector are then roughly Gaussian distributed, with variance $\sigma^2 \cdot (nB^2 + 2^t)$. By running a distinguisher on the created samples $\mathbf{z}$ we can verify whether our initial guess is correct or not. After restoring some secret value, the whole procedure can be repeated, but for a smaller dimension.

D. Illustrations of Coded-BKW with Sieving

A micro picture of how coded-BKW with sieving works can be found in Figure 1. A sample gets mapped to the correct list $\mathcal{L}_{\Delta_i}$, based on the current n_i positions. Here $\mathcal{L}_{\Delta_1}, \ldots, \mathcal{L}_{\Delta_K}$ denote the set of all the K such lists. In this list $\mathcal{L}_{\Delta_i}$, when adding/subtracting two vectors the resulting vector gets elements that are on average smaller than B in magnitude in the current n_i positions. Then we only add/subtract vectors in the list $\mathcal{L}_{\Delta_i}$ such that the elements of the resulting vector on average is smaller than B in the first N_i positions. The list of such vectors is then denoted $\mathcal{S}_{\Delta_i}$.

A macro picture of how coded-BKW with sieving works can be found in Figure 2. The set of all samples gets divided up into lists $\mathcal{L}_{\Delta_1}, \ldots, \mathcal{L}_{\Delta_K}$. Sieving is then applied to each individual list. The resulting sieved lists $\mathcal{S}_{\Delta_1}, \ldots, \mathcal{S}_{\Delta_K}$ then constitute the set of samples for the next step of coded-BKW with sieving.

E. High-Level Comparison with Previous BKW Versions

A high-level comparison between the behaviors of plain BKW, coded-BKW and coded-BKW with sieving is shown in Figure 3.

Initially the average norm of all elements in a sample vector $\mathbf{a}$ is around $q/4$, represented by the first row in the figure. Plain BKW then gradually works towards a zero vector by adding/subtracting vectors in each step such that a fixed number of positions gets canceled out to 0.

The idea of coded-BKW is to not cancel out the positions completely, and thereby allow for longer steps. The positions that are not canceled out increase in magnitude by a factor of $\sqrt{2}$ in each step. To end up with an evenly distributed noise vector in the end we can let the noise in the new almost canceled positions increase by a factor of $\sqrt{2}$ in each step. Thus we can gradually increase the step size.

When reducing positions in coded-BKW, the previously reduced positions increase in magnitude by a factor of $\sqrt{2}$. However, the sieving step in coded-BKW with sieving makes sure that the previously reduced positions do not increase in magnitude. Thus, initially, we do not have to reduce the positions as much as in coded-BKW. However, the sieving

[6]For finding the shortest vector in a lattice, the problem gets easier using this assumption. However, in our case, if the distribution of vectors is biased, it gets easier to find pairs of vectors close to each other.

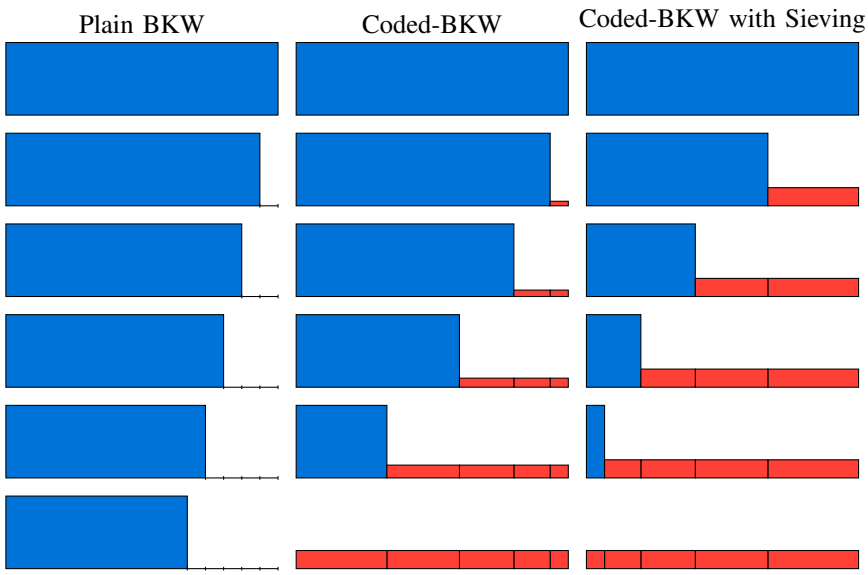

Fig. 3. A high-level illustration of how the different versions of the BKW algorithm work. The x-axis represents positions in the $\mathbf{a}$ vector, and the y-axis depicts the average absolute value of the corresponding position. The blue color corresponds to positions that have not been reduced yet and the red color corresponds to reduced positions.

process gets more expensive the more positions we work with, and we must therefore gradually divide our samples into fewer buckets to not increase the total cost of the later steps. Thus, we must gradually decrease the step size.

VI. Parameter Selection and Asymptotic Analysis

After each step, positions that already have been treated should remain at some given magnitude B. That is, the average (absolute) value of a treated position should be very close to B. This property is maintained by the way in which we apply the sieving part at each reduction step. After t steps we have therefore produced vectors of average norm $\sqrt{n} \cdot B$.

Assigning the number of samples to be $m = 2^k$, where 2^k is a parameter that will decide the total complexity of the algorithm, we will end up with roughly $m = 2^k$ samples after t steps. As already stated, these received samples will be roughly Gaussian with variance $\sigma^2 \cdot (nB^2 + 2^t)$. We assume that the best strategy is to keep the magnitudes of the two different contributions of the same order, so we choose $nB^2 \approx 2^t$.

Furthermore, using Equation (1), in order to be able to recover a single secret position using m samples, we need

$$m = \mathcal{O}\left(e^{4\pi^2 \cdot \frac{\sigma^2 \cdot (nB^2+2^t)}{q^2}}\right).$$

Thus, we have

$$\ln 2 \cdot k = 4\pi^2 \cdot \frac{\sigma^2 \cdot (nB^2 + 2^t)}{q^2} + \mathcal{O}(1). \tag{6}$$

Each of the t steps should deliver $m = 2^k$ vectors of the form described before.

Since we have two parts in each reduction step, we need to analyze these parts separately. First, consider performing the first part of reduction step number i using coded-BKW with an $[n_i, d_i]$ linear code, where the parameters n_i and d_i at each step are chosen for optimal (global) performance. We sort the 2^k vectors into $K = \frac{q^{d_i}-1}{2}$ different lists. Here the coded-BKW step guarantees that all the vectors in a list, restricted to the n_i considered positions, have an average norm less than $\sqrt{n_i} \cdot B$ if the codeword is subtracted from the vector. So the number of lists $\frac{q^{d_i}-1}{2}$ has to be chosen so that this norm restriction is true. Then, after the coded-BKW step, the sieving step should leave the average norm over the N_i positions unchanged, i.e., less than $\sqrt{N_i} \cdot B$.

Since all vectors in a list can be considered to have norm $\sqrt{N_i} \cdot B$ in these N_i positions, the sieving step needs to find any pair that leaves a difference between two vectors of norm at most $\sqrt{N_i} \cdot B$. Using the heuristic that the vectors are uniformly distributed on a sphere with radius $\sqrt{N_i} \cdot B$, we know that a single list should contain at least $2^{0.208N_i}$ vectors to be able to produce the same number of vectors. The time and space complexity is $2^{0.292N_i}$ if LSF is employed.

Let us adopt some further notation. As we expect the number of vectors to be exponential we write $k = c_0 n$ for some c_0. Also, we adopt $q = n^{c_q}$ and $\sigma = n^{c_s}$. By choosing $nB^2 \approx 2^t$, from (6) we derive that

$$B = \Theta(n^{c_q - c_s}) \tag{7}$$

and

$$t = (2(c_q - c_s) + 1)\log_2 n + \mathcal{O}(1). \tag{8}$$

A. Asymptotics of Coded-BKW with Sieving

We assume exponential overall complexity and write it as 2^{cn} for some coefficient c to be determined. Each step is additive with respect to complexity, so we assume that we can use 2^{cn} operations in each step. In the t steps we are choosing $n_1, n_2, \ldots$ positions for each step.

The number of buckets needed for the first step of coded-BKW is $(C' \cdot n^{c_s})^{n_1}$, where C' is another constant. In each

bucket the dominant part in the time complexity is the sieving cost $2^{\lambda n_1}$, for a constant λ. The overall complexity, the product of these expressions, should match the bound 2^{cn}, and thus we choose n_1 such that $(C' \cdot n^{c_s})^{n_1} \approx 2^{cn} \cdot 2^{-\lambda n_1}$.

Taking the log, $c_s \log n \cdot n_1 + \log C' n_1 = cn - \lambda n_1$. Therefore, we obtain

$$n_1 = \frac{cn}{c_s \log n + \lambda + \log C'}.$$

To simplify expressions, we use the notation $W = c_s \log n + \lambda + \log C'$.

For the next step, we get $W \cdot n_2 = cn - \lambda n_1$, which simplifies in asymptotic sense to

$$n_2 = \frac{cn}{W}\left(1 - \frac{\lambda}{W}\right).$$

Continuing in this way, we have $W \cdot n_i = cn - \lambda \sum_{j=1}^{i-1} n_j$ and we can obtain an asymptotic expression for n_i as

$$n_i = \frac{cn}{W}\left(1 - \frac{\lambda}{W}\right)^{i-1}.$$

After t steps we have $\sum_{i=1}^{t} n_i = n$, so we observe that

$$\sum_{i=1}^{t} n_i = \frac{cn}{W} \sum_{i=1}^{t} \left(1 - \frac{\lambda}{W}\right)^{i-1},$$

which simplifies to

$$n = \sum_{i=1}^{t} n_i = \frac{cn}{\lambda}\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t}\right).$$

Now, we know that

$$c = \lambda \left(1 - \left(1 - \frac{\lambda}{W}\right)^{t}\right)^{-1}.$$

Since t and W are both of order $\Theta(\log n)$ that tend to infinity as n tends to infinity, we have that

$$c = \lambda \left(1 - \left(1 - \frac{\lambda}{W}\right)^{\frac{W}{\lambda} \cdot \frac{t\lambda}{W}}\right)^{-1} \to \lambda \left(1 - e^{-\frac{t\lambda}{W}}\right)^{-1},$$

when $n \to \infty$.

Since $t/W \to (1 + 2(c_q - c_s))/c_s$ when $n \to \infty$ this finally gives us

$$c = \frac{\lambda}{1 - e^{-\lambda(1+2(c_q-c_s))/c_s}}.$$

Now assume the sieving heuristic, that after step i, the vectors restricted to the first N_i positions are uniformly distributed on a sphere with radius $\sqrt{N_i} \cdot B$. Also assume that the discrete Gaussian distributions can be approximated by continuous ones. Then we obtain the following theorem.

Theorem 1: The time and space complexity of the proposed algorithm is $2^{(c+o(1))n}$, where

$$c = \frac{\lambda}{1 - e^{-\lambda(1+2(c_q-c_s))/c_s}},$$

and $\lambda = 0.292$ for classic computers and 0.265 for quantum computers.

Proof: Since $c > \lambda$, there are exponential samples left for the distinguishing process. One can adjust the constants in (7) and (8) to ensure a success probability of hypothesis testing close to 1. ■

B. Asymptotics when Using Plain BKW Pre-Processing

In this section we show that Theorem 1 can be improved for certain LWE parameters. Suppose that we perform t_0 plain BKW steps and t_1 steps of coded-BKW with sieving, so $t = t_0 + t_1$. We first derive the following lemma.

Lemma 1: It is asymptotically beneficial to perform t_0 plain BKW steps, where t_0 is of order $(2(c_q - c_s) + 1 - c_s/\lambda \cdot \ln(c_q/c_s)) \log n$, if

$$\frac{c_s}{\lambda} \ln \frac{c_q}{c_s} < 2(c_q - c_s) + 1.$$

Proof: Suppose in each plain BKW step, we zero-out b positions. Therefore, we have that

$$q^b = 2^{cn+o(n)},$$

and it follows that asymptotically

$$b = \frac{cn}{c_q \log n} + o\left(\frac{n}{\log n}\right). \tag{9}$$

Because the operated positions in each step will decrease using coded-BKW with sieving, it is beneficial to replace a step of coded-BKW with sieving by a pre-processing step of plain BKW, if the allowed number of steps is large. We compute t_1 such that for $t \geq i \geq t_1$, we have $n_i \leq b$. That is,

$$\frac{cn}{W}\left(1 - \frac{\lambda}{W}\right)^{t_1 - 1} = \frac{cn}{c_q \log n}.$$

Thus, we derive that t_1 is of order $c_s/\lambda \cdot \ln(c_q/c_s) \cdot \log n$. ■

If we choose $t_0 = t - t_1$ plain BKW steps, where t_1 is of order $c_s/\lambda \cdot \ln(c_q/c_s) \cdot \log n$ as in Lemma 2, then

$$n - t_0 b = \sum_{i=1}^{t_1} n_i = \frac{cn}{\lambda}\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t_1}\right).$$

Thus

$$1 - \frac{c}{c_q}\left(2(c_q - c_s) + 1 - \frac{c_s}{\lambda} \ln\left(\frac{c_q}{c_s}\right)\right) = \frac{c}{\lambda}\left(1 - \frac{c_s}{c_q}\right).$$

Finally, making the same heuristic assumptions as in Theorem 1, we have the following theorem for characterizing its asymptotic complexity.

Theorem 2: If $c > \lambda$ and $\frac{c_s}{\lambda} \ln \frac{c_q}{c_s} < 2(c_q - c_s) + 1$, then the time and space complexity of the proposed algorithm with plain BKW pre-processing is $2^{(c+o(1))n}$, where

$$c = \frac{\lambda c_q}{(1 + 2\lambda)(c_q - c_s) + \lambda - c_s \ln\left(\frac{c_q}{c_s}\right)},$$

and $\lambda = 0.292$ for classic computers and 0.265 for quantum computers.

Proof: The proof is similar to that of Theorem 1. ■

TABLE I
ASYMPTOTIC COMPLEXITY FOR THE REGEV PARAMETERS

Algorithm	Complexity exponent (c)
QS-BKW(w/ p)	0.8856
S-BKW(w/ p)	0.8951
S-BKW(w/o p)	0.9054
Coded-BKW [25], [31]	0.9299
DUAL-POLYSamples [27]	4.6720
DUAL-EXPSamples [27]	1.1680

C. Case Study: Asymptotic Complexity of the Regev Parameters

In this part we present a case-study on the asymptotic complexity of Regev parameter sets, a family of LWE instances with significance in public-key cryptography.

Regev parameters: We pick parameters $q \approx n^2$ and $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$ as suggested in [45].

The asymptotic complexity of Regev's LWE instances is shown in Table I. For this parameter set, we have $c_q = 2$ and $c_s = 1.5$, and the previously best algorithms in the asymptotic sense are the coded-BKW variants [25], [31] (denoted Coded-BKW in this table) with time complexity $2^{0.9299n+o(n)}$. The item DUAL-POLYSamples represents the run time exponent of lattice reduction approaches using polynomial samples and exponential memory, while DUAL-EXPSamples represents the run time exponent of lattice reduction approaches using exponential samples and memory. Both values are computed according to formulas from [27], i.e., $2c_{BKZ} \cdot c_q/(c_q - c_s)^2$ and $2c_{BKZ} \cdot c_q/(c_q - c_s + 1/2)^2$, respectively. Here c_{BKZ} is chosen to be 0.292, the best constant that can be achieved heuristically [9].

We see from the table that the newly proposed algorithm coded-BKW with sieving outperforms the previous best algorithms asymptotically. For instance, the simplest strategy without plain BKW pre-processing, denoted S-BKW(w/o p), costs $2^{0.9054n+o(n)}$ operations, with pre-processing, the time complexity, denoted S-BKW(w/ p) is $2^{0.8951n+o(n)}$. Using quantum computers, the constant hidden in the exponent can be further reduced to 0.8856, shown in Table I as QS-BKW(w/ p). Note that the exponent of the lattice approach is much higher than that of the BKW variants for the Regev parameters.

D. A Comparison with the Asymptotic Complexity of Other Algorithms

A comparison between the asymptotic time complexity of coded-BKW with sieving and the previous best single-exponent algorithms is shown in Figure 4, similar to the comparison made in [28]. The upper and lower picture show the state-of-the-art algorithms before and after coded-BKW with sieving was introduced. We use pre-processing with standard BKW steps (see Theorem 2), since that reduces the complexity of the coded-BKW with sieving algorithm for the entire plotted area. Use of exponential space is assumed. Access to an exponential number of samples is also assumed.

First of all we notice that coded-BKW with sieving beats coded-BKW for all the parameters in the figure. It also outperforms the dual algorithm with an exponential number of samples on some areas where that algorithm used to be the best. It is also worth mentioning that the Regev instances are well within the area where coded-BKW with sieving performs best.

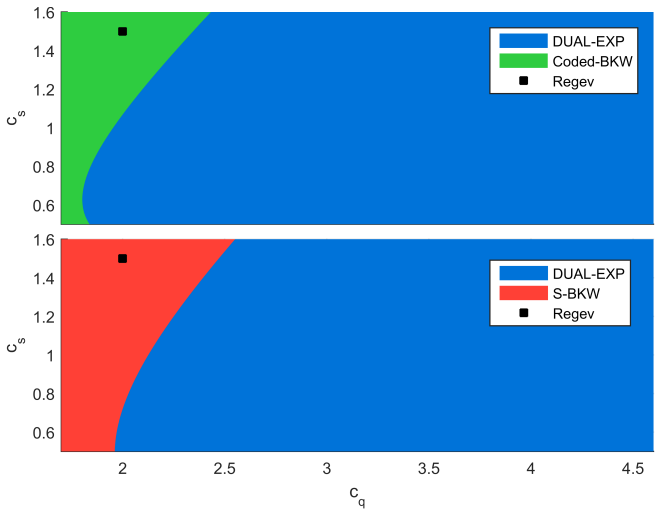

Fig. 4. A comparison of the asymptotic behavior of the best single-exponent algorithms for solving the LWE problem for different values of c_q and c_s. The different areas show where in the parameter space the corresponding algorithm beats the other algorithms in that subplot.

We have omitted the area where $c_s < 0.5$ in Figure 4 and the subsequent figures. Here the Arora-Ge algorithm [8] is polynomial. This area is not particularly interesting in cryptographical terms, since Regev's reduction proof does not apply for $c_s < 0.5$.

VII. ASYMPTOTIC COMPLEXITY OF LWE WITH SPARSER SECRETS

In this part, we discuss the asymptotic solving complexity of an LWE variant whose secret symbols are sampled from a distribution with standard deviation $n^{c_{s_1}}$ and the error distribution is a discrete Gaussian with standard deviation $n^{c_{s_2}}$, where $0 < c_{s_1} < c_{s_2}$. We make the same heuristic assumptions as in Theorem 1 in the derivations in this section. One important application is the LWE problem with a polynomial number of samples, where c_{s_1} equals c_s, while c_{s_2} changes to

$$\begin{cases} c_s + \frac{1}{2} & \text{if we start with } \Theta(n \log n) \text{ samples,} \\ c_s + \frac{1}{2} + \frac{c_q}{c_m - 1} & \text{if we start with } \Theta(c_m n) \text{ samples,} \end{cases}$$

after the secret-noise transform and the sample amplification procedure (cf. [28]).

We assume that the best strategy is to choose $nB^2 n^{2c_{s_1}} \approx 2^t n^{2c_{s_2}}$. Therefore, we know that $B = C \cdot n^{c_q - c_{s_1}}$ and $t = \log_2 D + (2(c_q - c_{s_2}) + 1) \cdot \log_2 n$, for some constants C and D. We then derive similar formulas except that now $W = c_{s_1} \cdot \log n + o(\log n)$.

We have the following theorem.

Theorem 3: The time and space complexity of the proposed algorithm for solving the LWE problem with sparse secrets is $2^{(c+o(1))n}$, where

$$c = \frac{\lambda}{1 - e^{-\lambda\left(1+2\left(c_q - c_{s_2}\right)\right)/c_{s_1}}},$$

and $\lambda = 0.292$ for classic computers and 0.265 for quantum computers.

Lemma 2: It is asymptotically beneficial to perform t_0 plain BKW steps, where t_0 is of order $\left(2\left(c_q - c_{s_1}\right) + 1 - c_{s_1}/\lambda \cdot \ln\left(c_q/c_{s_1}\right)\right) \log n$, if

$$\frac{c_{s_1}}{\lambda} \ln \frac{c_q}{c_{s_1}} < 2\left(c_q - c_{s_1}\right) + 1.$$

If we choose $t_0 = t - t_1$ plain BKW steps, where t_1 is of order $c_{s_1}/\lambda \cdot \ln\left(c_q/c_{s_1}\right) \cdot \log n$ as in Lemma 2, then

$$n - t_0 b = \sum_{i=1}^{t_1} n_i = \frac{cn}{\lambda}\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t_1}\right).$$

Thus

$$1 - \frac{c}{c_q}\left(2\left(c_q - c_{s_2}\right) + 1 - \frac{c_{s_1}}{\lambda} \ln\left(\frac{c_q}{c_{s_1}}\right)\right) = \frac{c}{\lambda}\left(1 - \frac{c_{s_1}}{c_q}\right).$$

Finally, we have the following theorem.

Theorem 4: If $c > \lambda$ and $\frac{c_{s_1}}{\lambda} \ln \frac{c_q}{c_{s_1}} < 2\left(c_q - c_{s_1}\right) + 1$, then the time and space complexity of the proposed algorithm with plain BKW pre-processing for solving the LWE problem with sparse secrets is $2^{(c+o(1))n}$, where c is

$$\frac{\lambda c_q}{\left(c_q - c_{s_1}\right) + \lambda\left(2\left(c_q - c_{s_2}\right) + 1\right) - c_{s_1} \ln\left(\frac{c_q}{c_{s_1}}\right)},$$

and $\lambda = 0.292$ for classic computers and 0.265 for quantum computers.

A. Asymptotic Complexity of LWE with a Polynomial Number of Samples

Applying Theorems 3 and 4 to the case where we limit the number of samples to $\Theta(n \log n)$ gives us the comparison of complexity exponents for the Regev parameters in Table II. Notice that, asymptotically speaking, the BKW algorithms perform much better compared to the lattice reduction counterparts in this scenario. Also, since pre-processing with plain BKW steps does not lower the complexity in the polynomial case, we just call the algorithms S-BKW and QS-BKW.

In Figure 5 we compare the asymptotic behavior between the different algorithms for varying values of c_q and c_s, when the number of samples is limited to $\Theta(n \log n)$. The upper and lower picture show the state-of-the-art algorithms before and after coded-BKW with sieving was introduced. Notice here that the area where the BKW algorithms perform better is much larger than in the case with exponential number of samples. Coded-BKW is best only in a very small area.

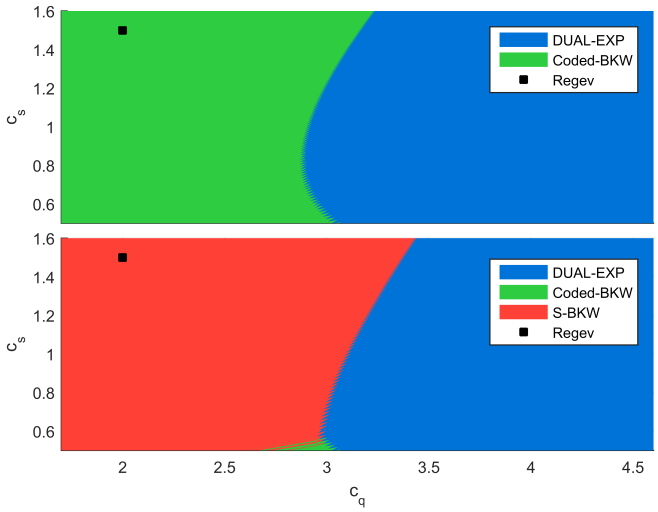

Fig. 5. A comparison of the asymptotic behavior of the best single-exponent algorithms for solving the LWE problem for different values of c_q and c_s. The different areas show where in the parameter space the corresponding algorithm beats the other algorithms in that subplot. The number of samples is limited to $\Theta(n \log n)$.

VIII. New Variants of Coded-BKW with Sieving

We come back to the general LWE problem (without a limit on the number of samples). In this section, we present three novel variants, the first two showing unified views for the existing BKW algorithms, and the other improving the asymptotic complexity for solving many LWE instances including the important Regev ones, both classically and in a quantum setting. We make the same heuristic assumptions as in Theorem 1 in the derivations in this section.

The results can easily be extended to the solving of LWE problems with sparse secrets, by replacing c_{s_2} below by $c_s + 1/2$, like we did in Section VII. The improvements in the sparse case are similar to the ones we show below, so for ease of reading we omit this analysis.

To balance the noise levels for the best performance, we always perform $t = (2(c_q - c_s) + 1) \log_2 n + \mathcal{O}(1)$ reduction steps and make the noise in each position approximately equal to $B = \Theta(n^{c_q - c_s})$. For the t reduction steps, we have three different choices, i.e., plain BKW, coded-BKW, and coded-BKW with sieving. We assume that plain BKW steps (if performed) should be done before the other two options[7].

A. S-BKW-v1

We start with a simple procedure (named S-BKW-v1), i.e., first performing t_1 plain BKW steps, then t_2 coded-BKW

[7]Assume that we apply coded-BKW or coded-BKW with sieving to the first steps and then plain BKW steps to the last steps. The noise corresponding to the first positions would then increase by a factor of $\sqrt{2}$ for each plain BKW step we take. Reversing the order, performing the plain BKW steps first and finishing with the coded-BKW/coded-BKW with sieving steps, we do not see the same increase in noise.

TABLE II
ASYMPTOTIC COMPLEXITY FOR THE REGEV PARAMETERS WITH A POLYNOMIAL NUMBER OF SAMPLES

Algorithm	Complexity exponent (c)
QS-BKW	1.6364
S-BKW	1.6507
Coded-BKW [25], [31]	1.7380
DUAL-POLYSamples [27]	4.6720

steps, and finally t_3 steps of coded-BKW with sieving. Thus,

$$t_1 + t_2 + t_3 = t = (2(c_q - c_s) + 1) \log_2 n + \mathcal{O}(1),$$

and we denote that $t_3 = \alpha \log n + \mathcal{O}(1)$, $t_2 = \beta \log n + \mathcal{O}(1)$, and $t_1 = (2(c_q - c_s) + 1 - \alpha - \beta) \log n + \mathcal{O}(1)$. A straightforward constraint is that

$$0 \leq \alpha, \beta \leq \alpha + \beta \leq 2(c_q - c_s) + 1.$$

This is a rather generic algorithm as all known BKW variants, i.e., plain BKW, coded-BKW, coded-BKW with sieving (with or without plain BKW pre-processing), can be treated as specific cases obtained by tweaking the parameters t_1, t_2 and t_3.

We want to make the noise variances for each position equal, so we set

$$B_i = \frac{B}{\sqrt{2^{t_3+i}}},$$

for $i = 1, \ldots, t_2$, where $2B_i$ is the reduced noise interval after $(t_2 - i + 1)$-th coded-BKW steps.

Let m_i be the length of the $(t_2 - i + 1)$-th coded-BKW step. We have that,

$$\left(\frac{q}{B_i}\right)^{m_i} \approx 2^{cn},$$

which simplifies to

$$cn = m_i(c_s \log n + \frac{t_3 + i}{2} + C_0).$$

Thus,

$$m_i = \frac{cn}{(c_s + \frac{\alpha}{2}) \log n + \frac{i}{2} + C_1}, \tag{10}$$

where C_1 is another constant. We know that

$$\sum_{i=1}^{t_2} m_i = 2cn \cdot \ln \frac{c_s + \frac{\alpha+\beta}{2}}{c_s + \frac{\alpha}{2}} + o(n). \tag{11}$$

Let n_i be the length of the i-th step of coded-BKW with sieving, for $1 \leq i \leq t_3$. We derive that

$$n_i \approx \frac{cn}{c_s \log n} \exp(-\frac{i}{c_s \log n}\lambda).$$

Therefore,

$$\sum_{i=1}^{t_3} n_i = \frac{cn}{\lambda}(1 - \exp(-\frac{\alpha}{c_s}\lambda)) + o(n). \tag{12}$$

We then have the following theorem.

Theorem 5: One (c_q, c_s) LWE instance can be solved with time and memory complexity $2^{(c+o(1))n}$, where c is the solution to the following optimization problem

$$\begin{aligned} \textbf{minimize}_{\alpha,\beta} \quad & c(\alpha, \beta) = (\frac{2(c_q - c_s) + 1 - \alpha - \beta}{c_q} \\ & + 2 \ln \frac{c_s + \frac{\alpha+\beta}{2}}{c_s + \frac{\alpha}{2}} + \lambda^{-1}(1 - \exp(-\frac{\alpha}{c_s}\lambda)))^{-1} \\ \textbf{subject to} \quad & 0 \leq \alpha, \beta \leq 2(c_q - c_s) + 1, \\ & \alpha + \beta \leq 2(c_q - c_s) + 1. \end{aligned}$$

Proof: Since $n = t_1 b + \sum_{i=1}^{t_2} m_i + \sum_{i=1}^{t_3} n_i$, we have

$$\begin{aligned} 1 =& c(\frac{2(c_q - c_s) + 1 - \alpha - \beta}{c_q} + 2 \ln \frac{c_s + \frac{\alpha+\beta}{2}}{c_s + \frac{\alpha}{2}} \\ & + \lambda^{-1}(1 - \exp(-\frac{\alpha}{c_s}\lambda))), \end{aligned}$$

where b is obtained from (9). ∎

Example 1: For the Regev parameters, i.e., $(c_q, c_s) = (2, 1.5)$, we derive that $\beta = 0$ for the best asymptotic complexity of S-BKW-v1. Thus, in this scenario, this generic procedure degenerates to coded-BKW with sieving using plain BKW processing discussed in Section VI-B, i.e., including no coded-BKW steps.

B. S-BKW-v2

Next, we present a variant (named S-BKW-v2) of coded-BKW with sieving by changing the order of the different BKW reduction types in S-BKW-v1. We first do t_1 plain BKW steps, then t_2 coded-BKW with sieving steps, and finally t_3 coded-BKW steps. Similarly, we let $t_3 = \alpha \log n + \mathcal{O}(1)$, $t_2 = \beta \log n + \mathcal{O}(1)$, and $t_1 = (2(c_q - c_s) + 1 - \alpha - \beta) \log n + \mathcal{O}(1)$. We also have the constraint

$$0 \leq \alpha, \beta \leq \alpha + \beta \leq 2(c_q - c_s) + 1.$$

This is also a generic framework including all known BKW variants as its special cases.

Let m_i represent the length of the $(t_3 - i + 1)$-th coded-BKW step, for $1 \leq i \leq t_3$, and n_j the length of the j-th step of coded-BKW step with sieving, for $1 \leq j \leq t_2$.

We derive that,

$$m_1 = \frac{cn}{c_s \log n} + o\left(\frac{n}{\log n}\right),$$

$$m_{t_3} = \frac{cn}{\left(c_s + \frac{\alpha}{2}\right) \log n} + o\left(\frac{n}{\log n}\right),$$

$$\sum_{i=1}^{t_3} m_i = 2cn \cdot \ln \frac{c_s + \frac{\alpha}{2}}{c_s} + o(n),$$

$$n_{t_2} = \frac{cn}{(c_s + \frac{\alpha}{2}) \log n} \exp\left(-\frac{\beta}{c_s}\lambda\right) + o\left(\frac{n}{\log n}\right),$$

$$\sum_{j=1}^{t_2} n_j = \frac{cc_s \cdot n}{\lambda(c_s + \frac{\alpha}{2})} \left(1 - \exp\left(-\frac{\beta\lambda}{c_s}\right)\right) + o(n).$$

We then have the following theorem.

Theorem 6: One (c_q, c_s) LWE instance can be solved with time and memory complexity $2^{(c+o(1))n}$, where c is the solution to the following optimization problem

$$\begin{aligned} \underset{\alpha,\beta}{\textbf{minimize}} \quad & c(\alpha,\beta) = \left(\frac{c_s}{\lambda\left(c_s + \frac{\alpha}{2}\right)}\left(1 - \exp\left(-\frac{\beta\lambda}{c_s}\right)\right) + \right. \\ & \left. 2\ln\frac{c_s + \frac{\alpha}{2}}{c_s} + \frac{1}{c_q}\left(2\left(c_q - c_s\right) + 1 - \alpha - \beta\right)\right)^{-1} \\ \textbf{subject to} \quad & 0 \le \alpha, \beta \le 2(c_q - c_s) + 1, \\ & \alpha + \beta \le 2(c_q - c_s) + 1. \end{aligned}$$

Proof: The proof is similar to that of Theorem 5. ∎

Example 2: For Regev parameters, we derive that $\alpha = 0$ for the best asymptotic complexity of S-BKW-v2, so it also degenerates to coded-BKW with sieving using plain BKW processing discussed in Section VI-B.

C. S-BKW-v3

We propose a new variant (named S-BKW-v3) including a nearest neighbor searching algorithm after a coded-BKW step, which searches for a series of new vectors whose norm is smaller with a factor of γ, where $0 \le \gamma \le \sqrt{2}$, by adding or subtracting two vectors in a ball. This is a generalization of coded-BKW and coded-BKW with sieving from another perspective, since coded-BKW can be seen as S-BKW-v3 with reduction parameter $\gamma = \sqrt{2}$, and coded-BKW with sieving as S-BKW-v3 with reduction parameter $\gamma = 1$.

We denote the complexity exponent for the nearest neighbor searching algorithm λ, i.e., $2^{\lambda n + o(n)}$ time and space is required if the dimension is n. We can improve the asymptotic complexity for the Regev parameters further.

We start by performing t_1 plain BKW steps and then t_2 steps of coded-BKW with sieving using parameters (λ, γ). Let $t_2 = \alpha \log n + \mathcal{O}(1)$ and $t_1 = t - t_2 = (2(c_q - c_s) + 1 - \alpha)\log n + \mathcal{O}(1)$. We also have the constraint that $0 \le \alpha \le 2(c_q - c_s) + 1$.

Let n_1 be the length of the first step of coded-BKW with sieving. We have $B_1 = B/\gamma^{t_2}$, so

$$(t_2 \log\gamma + c_s \log n) \cdot n_1 = cn - \lambda n_1.$$

Thus,

$$\begin{aligned} n_1 &= \frac{cn}{(c_s + \alpha\log\gamma)\log n + C} \\ &= \frac{cn}{(c_s + \alpha\log\gamma)\log n} \cdot \left(1 + \Theta\left(\log^{-1} n\right)\right), \end{aligned}$$

where C is a constant.

For the i-th step of coded-BKW with sieving, we derive that

$$((t_2 - i + 1)\log\gamma + c_s \log n) \cdot n_i = cn - \lambda \sum_{j=1}^{i} n_j. \quad (13)$$

Thus,

$$n_i = \left(1 + \frac{\log\gamma - \lambda}{(t_2 - i + 1)\log\gamma + c_s \log n + \lambda}\right) \cdot n_{i-1}$$

and if $\gamma \ne 1$, we have that

$$\begin{aligned} n_{t_2} &= \prod_{i=2}^{t_2}\left(1 + \frac{\log\gamma - \lambda}{(t_2 - i + 1)\log\gamma + c_s \log n + \lambda}\right) \cdot n_1 \\ &= n_1 \cdot \exp\left(\sum_{i=2}^{t_2} \ln\left(\frac{\log\gamma - \lambda}{(t_2 - i + 1)\log\gamma + c_s \log n + \lambda} \right.\right. \\ &\quad \left.\left. +1\right)\right) \\ &= n_1 \cdot \exp\left(\sum_{i=2}^{t_2}\left(\frac{\log\gamma - \lambda}{(t_2 - i + 1)\log\gamma + c_s \log n + \lambda} \right.\right. \\ &\quad \left.\left. +\Theta\left(\log^{-2} n\right)\right)\right) \\ &= n_1 \cdot \exp\left(\int_0^{\alpha} \frac{\log\gamma - \lambda}{t\log\gamma + c_s} dt + \Theta\left(\log^{-1} n\right)\right) \\ &= n_1 \cdot \exp\left(\frac{\log\gamma - \lambda}{\log\gamma} \cdot \ln\frac{c_s + \alpha\log\gamma}{c_s} + \Theta\left(\log^{-1} n\right)\right) \\ &= \frac{n}{\log n} \cdot \frac{c}{c_s + \alpha\log\gamma} \exp\left(\frac{\log\gamma - \lambda}{\log\gamma} \cdot \ln\frac{c_s + \alpha\log\gamma}{c_s}\right) \\ &\quad +o\left(\frac{n}{\log n}\right). \end{aligned}$$

We also know that,

$$N = \sum_{j=1}^{t_2} n_j = \lambda^{-1}\left(cn - (\log\gamma + c_s \log n) \cdot n_{t_2}\right)$$

Thus,

$$N = \lambda^{-1}\left(cn - \left(\frac{c_s}{\alpha\log\gamma + c_s}\right)^{\frac{\lambda}{\log\gamma}} \cdot cn + o(n)\right). \quad (14)$$

If $t_1 b + N = n$, then the following equation holds,

$$\begin{aligned} n = & \quad (2(c_q - c_s) + 1 - \alpha)\frac{cn}{c_q} \\ & + \frac{cn}{\lambda}\left(1 - \left(\frac{c_s}{\alpha\log\gamma + c_s}\right)^{\frac{\lambda}{\log\gamma}}\right). \end{aligned}$$

Thus, we derive the following formula to compute the constant c, i.e.

$$\left(2\left(1 - \frac{c_s}{c_q} + \frac{1-\alpha}{2c_q}\right) + \frac{1}{\lambda}\left(1 - \left(\frac{c_s}{\alpha\log\gamma + c_s}\right)^{\frac{\lambda}{\log\gamma}}\right)\right)^{-1}.$$

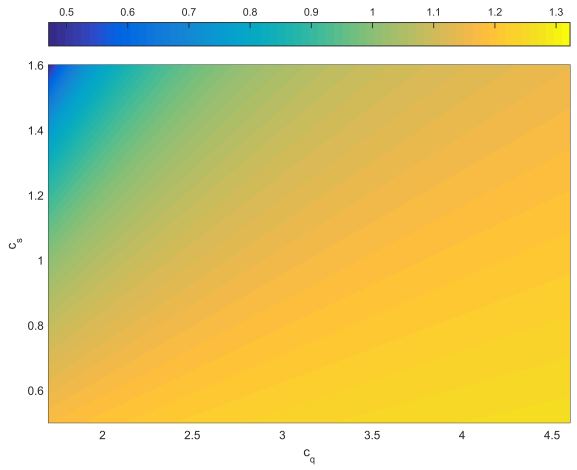

Fig. 6. The optimal γ value for version D of the algorithm, as a function of (c_q, c_s).

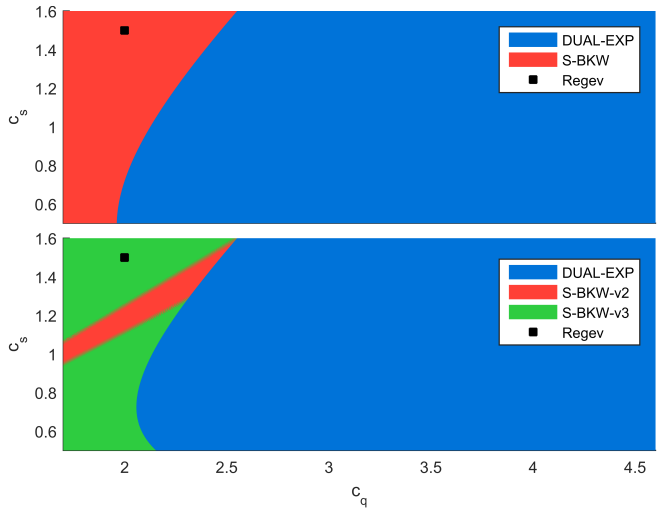

Fig. 7. A comparison of the asymptotic behavior of the best single-exponent algorithms for solving the LWE problem for different values of c_q and c_s. The different areas show where in the parameter space the corresponding algorithm beats the other algorithms in that subplot.

Theorem 7: If $0 \leq \alpha_0 \leq 2(c_q - c_s) + 1$ and $\gamma \neq 1$, then a (c_q, c_s) LWE instance can be solved with time and memory complexity $2^{(c+o(1))n}$, where c is

$$\left(\left(2\left(1-\frac{c_s}{c_q}\right)+\frac{1-\alpha_0}{c_q}\right)+\frac{1}{\lambda}\left(1-\left(\frac{c_s}{\alpha_0\log\gamma+c_s}\right)^{\frac{\lambda}{\log\gamma}}\right)\right)^{-1}. \tag{15}$$

Example 3: The numerical results for the Regev parameters using various reduction factors are listed in Table III, where the complexity exponent λ of the nearest neighbor searching is computed by the LSF approach [9]. Using S-BKW-v3, we can further decrease the complexity exponent c for solving the Regev LWE instance from 0.8951 to 0.8927 classically, and from 0.8856 to 0.8795 in a quantum setting. For these parameters, the choice of γ to achieve the best asymptotic complexity is 0.86 classically (or 0.80 using a quantum computer).

1) Optimal Choice of γ: The optimal choice of γ depends on the parameters c_q and c_s, illustrated in Figure 6. The gap between c_q and c_s is important, since this optimal γ value increases with c_q for a fixed c_s, and decreases with c_s when c_q is determined.

D. An Asymptotic Comparison of the New Variants

We present a comparison describing the asymptotic behavior of the best single-exponential algorithms for solving the LWE problems with varying (c_q, c_s) in Figure 7. The upper sub-plot includes all previous algorithms, which have been shown in Figure 4. S-BKW refers to coded-BKW with sieving with preprocessing, as defined in Section VI-B. We further add the new BKW variants from this section in the lower part.

Notice that all previous BKW variants are special cases of the three new algorithms, i.e., S-BKW-v1, S-BKW-v2, and S-BKW-v3, so in the lower sub-plot and for a particular pair of (c_q, c_s), the best algorithm is always among these three variants and DUAL-EXP. From this sub-plot, firstly, the area where DUAL-EXP wins becomes significantly smaller. Secondly, with respect to the area that the BKW variants win, S-BKW-v3 beats the other two in most pairs of parameters. S-BKW-v2 is the best algorithm in a thin strip, and by comparing with Figure 6, we notice that this strip corresponds to an area where the optimal γ value is close to 1. Therefore, for the pairs of (c_q, c_s) in this area, optimizing for γ does not help that much, and S-BKW-v2 is superior. S-BKW-v1 never wins for parameters considered in this graph.

E. A High Level Description

A high level comparison showing how the different new versions of coded-BKW with sieving work, similar to Figure 3, can be found in Figure 8. In all versions, pre-processing with plain BKW steps is excluded from the description.

In S-BKW-v1, we first take coded-BKW steps. This means longer and longer steps, and gradually increasing noise. Then we switch to coded-BKW with sieving steps. Here the steps get shorter and shorter since we have to apply sieving to an increasing number of previous steps.

In S-BKW-v2, we begin with shorter and shorter coded-BKW with sieving steps, keeping the noise of the positions low. Then we finish off with longer and longer coded-BKW steps. Here we do not apply sieving to the previously sieved positions, thus the added noise of these positions grow.

For S-BKW-v3 we have two versions; S-BKW-v3a and S-BKW-v3b. These are identical except that they use different reduction factors γ. In S-BKW-v3a, we use regular coded-BKW with sieving steps with a reduction factor $\gamma > 1$. The steps get shorter and shorter because we need to sieve more and more positions. The added noise is small in the beginning, but in each sieved position it becomes larger and larger for each step. S-BKW-v3b is the same, except that we use $\gamma < 1$. This means that the noise is large in the beginning, but gets smaller and smaller for each step.

TABLE III
THE COMPLEXITY EXPONENT c FOR VARIOUS REDUCTION FACTORS WHEN $(c_q, c_s) = (2, 1.5)$.

γ		0.78	0.80	0.82	0.84	0.86	0.88	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04
λ	classic	0.610	0.577	0.544	0.512	0.482	0.452	0.423	0.395	0.368	0.342	0.317	0.292	0.269	0.246
	quantum	0.574	0.541	0.509	0.478	0.448	0.419	0.391	0.364	0.338	0.313	0.289	0.265	0.243	0.221
c	classic	0.8933	0.8930	0.8928	0.8927	**0.8927**	0.8928	0.8930	0.8932	0.8936	0.8940	0.8946	**0.8951**	0.8959	0.8967
	quantum	0.8796	**0.8795**	0.8795	0.8797	0.8800	0.8805	0.8810	0.8817	0.8825	0.8835	0.8845	**0.8856**	0.8870	0.8884

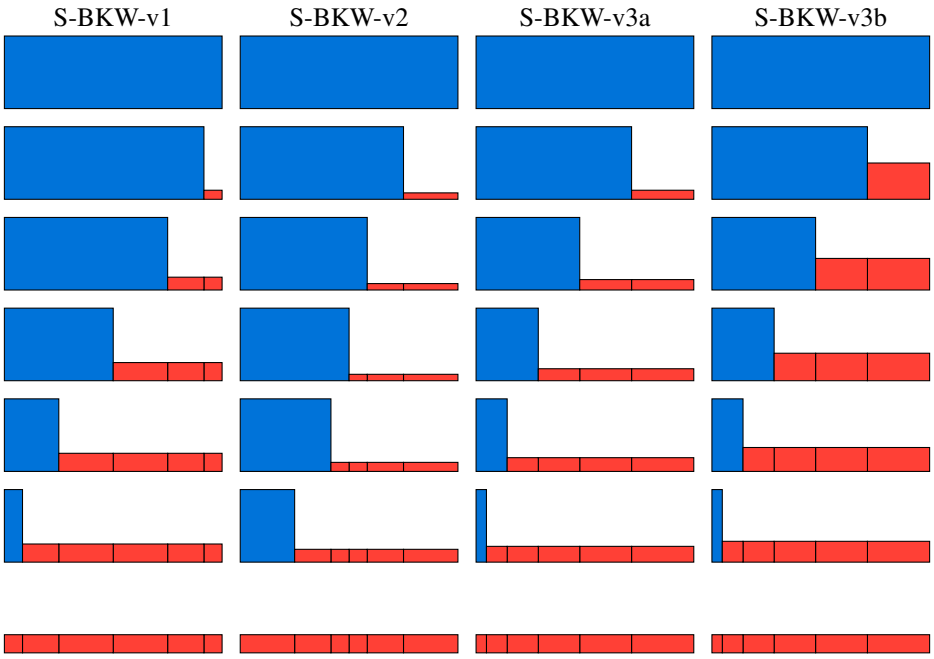

Fig. 8. A high-level illustration of how the different new variants of coded-BKW with sieving work. The x-axis represents positions in the $\mathbf{a}$ vector, and the y-axis depicts the average absolute value of the corresponding position. The blue color corresponds to positions that have not been reduced yet and the red color corresponds to reduced positions. Notice that the two rightmost columns both correspond to the same version of the algorithm, but with $\gamma > 1$ and $\gamma < 1$ respectively.

F. More Generalization

A straight-forward generalization of all the new variants described in Sections VIII-A-VIII-C is to allow different reduction parameter γ_i in different steps, after having pre-processed the samples with plain BKW steps. In addition, we can allow a sieving operation on positions in an interval $\mathcal{I}_i$ (or even more generally on any set of positions), using a flexible reduction factor γ_i. This optimization problem is complicated due to the numerous possible approaches, and generally, it is even difficult to write a closed formula for the objective function. We leave the problem of finding better asymptotic algorithms via extensive optimization efforts as an interesting scope for future research.

IX. CONCLUSIONS AND FUTURE WORK

In the paper we have presented a new BKW-type algorithm for solving the LWE problem. This algorithm, named coded-BKW with sieving, combines important ideas from two recent algorithmic improvements in lattice-based cryptography, i.e., coded-BKW and heuristic sieving for SVP, and outperforms the previously known approaches for important parameter sets in public-key cryptography.

For instance, considering Regev parameters, we have demonstrated an exponential asymptotic improvement, reducing the time and space complexity from $2^{0.930n}$ to $2^{0.893n}$. Additionally, we showed a similar improvement, when restricting the number of available samples to be polynomial. Lastly, we obtained the first quantum acceleration for this parameter set, further reducing the complexity to $2^{0.880n}$ if quantum computers are provided.

In the conference version [24], this algorithm has proven significant non-asymptotic improvements for some concrete parameters, compared with the previously best BKW variants. But one should further investigate the analysis when heuristics like unnatural selection[8] are taken into consideration, in order to fully exploit its power on suggesting accurate security parameters for real cryptosystems. Moreover, the influence on the concrete complexity of using varying reduction factors is unclear. For this purpose, further analysis and extensive simulation results are needed, which can be a very interesting topic for future work.

[8]Unnatural selection means creating more reduced vectors than needed and then choosing the best ones for the next step of the reduction. The idea is from [4].

Another stimulating problem is to search for new algorithms with better asymptotic complexity by solving the general optimization problem raised in Section VIII-F, numerically or analytically.

Lastly, the newly proposed algorithm definitely also has importance in solving many LWE variants with specific structures, e.g., the RING-LWE problem. An interesting research direction is to search for more applications, e.g., solving hard lattice problems, as in [31].

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers from ASIACRYPT 2017 and the reviewers for IEEE Transactions on Information Theory for their invaluable comments that helped improve the quality of this paper.

REFERENCES

[1] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing. pp. 601–610. ACM (2001)

[2] Albrecht, M., Cid, C., Faugere, J.C., Robert, F., Perret, L.: Algebraic algorithms for LWE problems. Cryptology ePrint Archive, Report 2014/1018 (2014)

[3] Albrecht, M.R., Cid, C., Faugere, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Designs, Codes and Cryptography 74(2), 325–354 (2015)

[4] Albrecht, M.R., Faugère, J.C., Fitzpatrick, R., Perret, L.: Lazy Modulus Switching for the BKW Algorithm on LWE. In: Krawczyk, H. (ed.) Public-Key Cryptography–PKC 2014, Lecture Notes in Computer Science, vol. 8383, pp. 429–445. Springer Berlin Heidelberg (2014), *http://dx.doi.org/10.1007/978-3-642-54631-0_25*

[5] Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the efficacy of solving LWE by reduction to unique-SVP. In: International Conference on Information Security and Cryptology. pp. 293–310. Springer (2013)

[6] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology 9(3), 169–203 (2015)

[7] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: Halevi, S. (ed.) Advances in Cryptology–CRYPTO 2009, Lecture Notes in Computer Science, vol. 5677, pp. 595–618. Springer Berlin Heidelberg (2009), *http://dx.doi.org/10.1007/978-3-642-03356-8_35*

[8] Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Automata, Languages and Programming, pp. 403–415. Springer (2011)

[9] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 10–24. Society for Industrial and Applied Mathematics (2016)

[10] Becker, A., Gama, N., Joux, A.: A sieve algorithm based on overlattices. LMS Journal of Computation and Mathematics 17(A), 49–70 (2014)

[11] Bernstein, D.J., Lange, T.: Never trust a bunny. In: Radio Frequency Identification. Security and Privacy Issues, pp. 137–148. Springer (2013)

[12] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant Learning, the Parity Problem, and the Statistical Query Model. In: Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing–STOC 2000, pp. 435–440. ACM (2000)

[13] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)

[14] Bogos, S., Vaudenay, S.: Optimization of LPN solving algorithms. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 703–728. Springer (2016)

[15] Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer (2012)

[16] Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106. IEEE Computer Society (2011)

[17] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Annual Cryptology Conference. pp. 505–524. Springer (2011)

[18] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Advances in Cryptology–ASIACRYPT 2011, pp. 1–20. Springer (2011)

[19] Conway, J. H., Sloane, N. J. A.: Sphere packings, lattices and groups. In: (Vol. 290). Springer Science and Business Media (2013)

[20] Dubiner, M.: Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. IEEE Transactions on Information Theory 56(8), 4166–4179 (2010)

[21] Duc, A., Tramèr, F., Vaudenay, S.: Better Algorithms for LWE and LWR. In: Advances in Cryptology – EUROCRYPT 2015, pp. 173–202. Springer (2015)

[22] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)

[23] Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Advances in Cryptology–ASIACRYPT 2014, pp. 1–20. Springer (2014)

[24] Guo, Q., Johansson, T., Mårtensson, E., Stankovski, P.: Coded-BKW with Sieving. In: Advances in Cryptology–ASIACRYPT 2017, Part I, pp. 323–346. Springer (2017)

[25] Guo, Q., Johansson, T., Stankovski, P.: Coded-BKW: Solving LWE using lattice codes. In: Advances in Cryptology–CRYPTO 2015, pp. 23–42. Springer (2015)

[26] Hanrot, G., Pujol, X., Stehlé, D.: Algorithms for the shortest and closest lattice vector problems. In: Coding and Cryptology, pp. 159–190. Springer (2011)

[27] Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving LWE. IACR Cryptology ePrint Archive 2015, 1222 (2015), *http://eprint.iacr.org/2015/1222*

[28] Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving LWE. J. Designs, Codes and Cryptography, pp. 1–29 (2017)

[29] Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 604–613. ACM (1998)

[30] Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011), *http://eprint.iacr.org/*

[31] Kirchner, P., Fouque, P.A.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: Advances in Cryptology–CRYPTO 2015, pp. 43–62. Springer (2015)

[32] Laarhoven, T.: Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In: Annual Cryptology Conference. pp. 3–22. Springer (2015)

[33] Laarhoven, T., Mosca, M., Van De Pol, J.: Finding shortest lattice vectors faster using quantum search. Designs, Codes and Cryptography 77(2-3), 375–400 (2015)

[34] Laarhoven, T., de Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In: International Conference on Cryptology and Information Security in Latin America. pp. 101–118. Springer (2015)

[35] Levieil, É., Fouque, P.A.: An improved LPN algorithm. In: Prisco, R.D., Yung, M. (eds.) SCN. Lecture Notes in Computer Science, vol. 4116, pp. 348–359. Springer-Verlag (2006)

[36] Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) Topics in Cryptology–CT-RSA 2011, Lecture Notes in Computer Science, vol. 6558, pp. 319–339. Springer Berlin Heidelberg (2011), *http://dx.doi.org/10.1007/978-3-642-19074-2_21*

[37] Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Topics in Cryptology–CT-RSA 2013, pp. 293–309. Springer (2013)

[38] May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. pp. 203–228 (2015)

[39] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput., pp. 372-381 (2004)

[40] Micciancio, D., Regev, O.: Lattice-based Cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer Berlin Heidelberg (2009)

[41] Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms. pp. 1468–1480. SIAM (2010)
[42] Mulder, E.D., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. J. Cryptographic Engineering 4(1), 33–45 (2014), *http://dx.doi.org/10.1007/s13389-014-0072-z*
[43] Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. Mathematical Cryptology 2(2), 181–207 (2008), *http://dx.doi.org/10.1515/JMC.2008.009*
[44] Pujol, X., Stehlé, D.: Solving the shortest lattice vector problem in time $2^{2.465n}$. IACR Cryptology ePrint Archive 2009, 605 (2009), *http://eprint.iacr.org/2009/605*
[45] Regev, O.: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Journal of the ACM 56(6), 34:1–34:40 (Sep 2009), *http://doi.acm.org/10.1145/1568318.1568324*
[46] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming 66(1-3), 181–199 (1994)
[47] Wagner, D.: A generalized birthday problem. In: Advances in cryptology–CRYPTO 2002, pp. 288–304. Springer (2002)
[48] Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. pp. 1–9. ACM (2011)
[49] Zhang, B., Jiao, L., Wang, M.: Faster algorithms for solving LPN. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 168–195. Springer (2016)
[50] Zhang, F., Pan, Y., Hu, G.: A three-level sieve algorithm for the shortest vector problem. In: International Conference on Selected Areas in Cryptography. pp. 29–47. Springer (2013)